논문번호 12-04-16
100

Journal of IKEEE.Vol.16,No.4,382~388,December 2012
http://dx.doi.org/10.7471/ikeee.2012.16.4.382

Advanced SIMD를 이용한 화면 간 예측 고속화방법

Acceleration Method of Inter Prediction using Advanced SIMD

김 완 수*, 이 재 흥*★

Wan-Su Kim*, Jae-Heung Lee*★

Abstract

An H.264/AVC fast motion estimation methodology is presented in this paper. Advanced SIMD based NEON which is one of the parallel processing methods is supported under the ARM Cortex-A9 dual-core platform. NEON is applied to a full search technique with one of the various motion estimation methods and SAD operation count of each macroblock is reduced to 1/4. Pixel values of the corresponding macroblock are assigned to eight 16-bit NEON registers and Intrinsic function in NEON architecture carried out 128 bits arithmetic operations at the same time. In this way, the exact motion vector with the minimum SAD value among the calculated SAD values can be designated. Experimental results show that performance gets improved 30% above average in accordance with the size of image and macroblock.

요 약

본 논문에서는 ARM Cortex-A9 듀얼코어 플랫폼에서 지원하는 병렬처리 기법 중 하나인 Advanced SIMD기반의 NEON을 적용한 H.264/AVC 고속화 움직임추정 방법론을 연구하였다. 다양한 움직임추정 방법 중 하나인 전역탐색기법에 NEON을 적용하여 각 매크로블록의 SAD 연산횟수를 1/4 감소시켰다. 해당 매크로블록의 픽셀 값들을 8개의 16bit NEON 레지스터에 할당하였고, NEON에서 지원하는 Intrinsic 함수를 사용하여 동시에 128bit 연산을 수행하였다. 이러한 방법으로 계산된 SAD 값들 중 최소 SAD 값을 가지고 정확한 모션벡터를 선정했다. 그 결과 영상의 크기 및 매크로블록의 크기에 따라 성능이 평균 30% 이상 향상되는 효과를 검증하였다.

Key words : H.264/AVC, Inter Prediction, Parallel Processing, NEON, SIMD

* Dept. of Computer Engineering, Hanbat University
ws8024@naver.com
Corresponding author

※ Acknowledgment
This research was financially supported by the Ministry of Education, Science Technology (MEST) and National Research Foundation of Korea(NRF) through the Human Resource Training Project for Regional Innovation
Manuscript received Dec. 9, 2012; revised Dec. 17, 2012; accepted Dec 17. 2012

Ⅰ. 서론

H.264/AVC는 ISO/IEC와 전기통신에 관한 국제표준화기구인 ITU-T(International Telecommunication Union-Telecommunication Standardization Sector, 국제전기통신연합-전기통신표준화부문)의 두 표준화기구가 비디오코덱 표준화 작업을 위해 JVT(Joint Video Team)를 조성하여 만든 비디오코덱이다[1]. 이 표준을 ITU-T 에서는 H.264 라고 하며 ISO/IEC 에

서는 MPEC-4 Part10 또는 AVC로 부르고 있다. 이전의 H.263과 MPEG-4 보다 일반적으로 50%이상 압축률을 향상시켰다. 반면에 다양한 압축 기술들의 추가/변경으로 인한 높은 복잡도를 요구하게 되었다. H.264/AVC가 앞으로 하드웨어의 발전을 예견하고 만든 것으로 기존의 코덱보다 같은 용량에서 선명한 영상을 제공하지만, 부호화/복호화에 걸리는 시간은 배 이상을 필요로 한다. 이런 복잡도의 문제 또는 사용 용도에 따른 문제를 해결하기 위한 방편으로 여러 가지의 프로파일과 레벨을 선택할 수 있다. 처음 베이스라인(Baseline), 메인(Main), 확장(Extended)의 세 가지 프로파일을 정의했으나 후에 CBP(Constrained Baseline Profile), 하이(High) 프로파일 등을 포함시켰다.

현존하는 비디오 코덱 중 가장 좋은 성능을 발휘하는 H.264/AVC는 많은 형태의 변환과정을 거쳐 최적화되고 우리 실생활 여러 곳에 사용되고 있다. TV나 모바일기기, 태블릿PC 등에서는 하드웨어 전용 칩셋을 사용하여 실시간으로 부호화 및 복호된 영상 데이터를 인터넷을 통해 스트리밍 하는 등 다양한 분야의 고부가가치 산업으로 발전하고 있는 추세이다. 이러한 추세를 반응하듯 아직까지도 용도에 맞는 최적화 작업이 활발히 이루어지고 있는 H.264/AVC에서 특히, ME(Motion Estimation)는 알고리즘 측면에서 가장 많은 접근이 이루어지는 부분 중 하나이다. 화면간 예측된 영상 데이터의 비트수와 연산 시간을 줄이기 위해 수많은 방법론들이 제안되었고 실제로 많은 성능향상 효과를 가져왔다. 하지만 이는 실행되는 플랫폼을 고려하지 않고 알고리즘적인 측면만을 고려한 것으로써 시스템 아키텍처적인 측면에서의 최적화는 아직 이루어지지 않은 부분들이 많아 반드시 최적화할 필요가 있다.

최근 모바일 기기들이 발전함에 따라 SoC(System On-Chip) 성능도 눈부시게 발전하고 있다. 그림 1은 ARM사에서 제공하는 Cortex-A9 프로세서로 ARM11 이 후 새롭게 선보인 차세대 마이크로프로세서이다. 기존 ARM 프로세서가 갖는 성능대비 전력소모에 대한 최적화를 이루었고, 다양한 기능들이 추가적으로 적용되어 저전력, 고성능 프로세서로 널리 알려져 있다. ARM Cortex-A 시리즈는 어플리케이션 마이크로프로세서로써 주로 모바일, 차량용 블랙박스, ECU, 셋톱박스, 네비게이션, PMP, 태블릿 PC 등의 시스템에 적용되어 사용된다[2]. 또한 Cortex-A8 프로세서부터 NEON이라는 미디어처리엔진[3]이 포함된다. NEON은 오디오, 비디오 및 2, 3차원 그래픽 프로세싱에 필요한 부동소수점 처리장치의 성능 및 기능

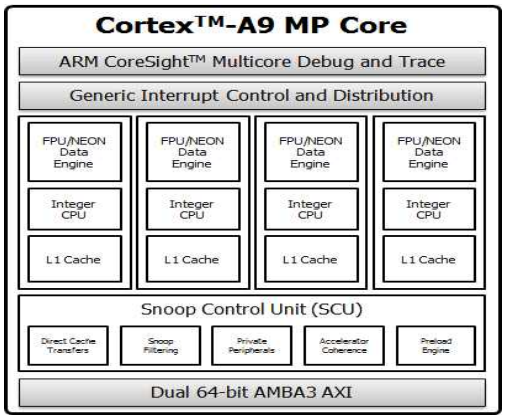

Fig. 1. Cortex-A9 MPCore Diagram with NEON
그림 1. NEON을 포함한 Cortex-A9 멀티코어 블록도

을 제공하는 것에 덧붙여 미디어 및 신호 처리 함수 작업을 가속화하기 위해 도입된 Advanced SIMD (Single Instruction Multiple Data) 아키텍처이다. NEON 미디어처리엔진은 효율적인 데이터 조작과 최소화환 메모리 액세스를 통해 데이터 처리성능을 향상시키는 장점을 갖고 있다. 본 논문에서는 이러한 시스템 아키텍처 접근을 통하여 H.264/AVC 중에서 가장 많은 시간과 연산량을 갖는 화면 간 예측 기술 중에서 ME를 고속화하여 처리할 수 있는 방법론을 제안한다.

본 논문의 구성은 다음과 같다. 2절에서 본 논문에서 핵심이 되는 NEON 기술과 H.264 ME에 대해 소개한다. 3절에서는 NEON 기술을 적용하여 구현한 고속화 ME를 소개하고, 4절에서 성능 측정 결과를 논의한다. 마지막으로 5절에서는 본 논문에 대한 결론 및 향 후 연구의 진행 방향을 제시한다.

II. 본론

2.1 Block-based Motion Estimation

움직임추정은 프레임 간의 시간적 상관성을 이용하여 움직임을 압축하는 기술이다. 이전 프레임과 현재 프레임 간의 시간적 유사성을 이용하여 움직임 정보를 추정하고 프레임을 생성한 뒤 생성된 프레임과 현재 프레임의 차이 값을 구한다. 움직임 정보와 두 프레임 간의 차이 값만 부호화하면 영상 정보 데이터를 크게 줄일 수 있다. 움직임추정은 IME(Integer-pel Motion Estimation)과 FME(Fractional-pel Motion Estimation)으로 구분된다. 영상에서 물체의 움직임은 정수 단위로만 일어나지 않기 때문에 정화소 단위에서의 움직임 벡터과 실제의 움직임벡터가 아닐 수도

있다. 이로 인해 H.264/AVC에서의 움직임추정은 IME를 통해 정화소 단위에서 수행하여 블록 매칭오차가 최소가 되는 지점을 중심으로 FME를 수행하여 최소 블록 매칭오차 지점을 찾는다[4].

최소 블록매칭 오차 지점을 측정하는 가장 일반적인 방법으로는 SAD(Sum of Absolute Difference) 방정식을 사용하며, 곱셈을 요구하지 않는 간편성 때문에 널리 사용된다. 현재 프레임을 MxN 크기로 분할하였을 때 (x, y)에 위치한 현재 매크로블록과 (x+i, y+j)에 위치한 블록 간의 SAD는 (식.1)로 표현할 수 있다.

$$AD\ i,j) = \sum_{=0\ y=0}^{-1\,N-1} F_t(x,y) - F_{t-1}(x,y)\Big| \quad (\quad .1)$$

여기에서 F_t는 현재 프레임을 나타내고, F_{t-1}은 이전 프레임을 나타낸다. 식 (1)을 통해 알 수 있듯이 해당되는 SAD 방정식의 해를 구하기 위해서는 MxN번의 뺄셈이 필요하다. 만약 탐색범위가 AxB일 때 전역탐색기법[5]을 사용한다면 매크로블록 한 개당 AxBxMxN 번의 뺄셈이 필요한데 이는 엄청난 연산량을 초래하고 CPU의 부하를 일으킬 수 있는 요소로 남을 수 있다. 움직임추정의 IME와 FME 과정은 영상 압축에서 일반적으로 연산량의 60% 이상을 차지하고 있는 중요한 처리과정 이므로 이러한 계산량을 줄이기 위한 연구가 아직까지도 활발하게 진행되고 있다.

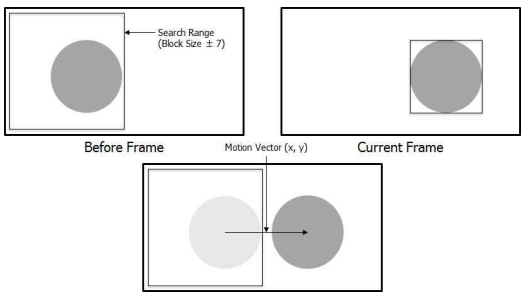

Fig. 2. Full Search Algorithm
그림 2. 전역탐색기법

그림 2와 같은 방법을 사용하는 전역탐색 알고리즘은 현재 프레임을 MxN 블록들로 나누고 정확한 움직임벡터를 얻기 위해 탐색영역의 후보 블록들을 주어진 범위 내에서 이전 프레임의 왼쪽 위에서부터 오른쪽 아래로 순서대로 현재 프레임의 탐색 영역(Search Range) 값과 비교한 후 최소 블록매칭 오차를 가진 블록의 위치를 움직임벡터로 지정한다. 각각의 블록에 대해서 정해진 탐색영역의 모든 위치에서 SAD 값들을 계산하여 SAD 값이 최소가 되는 위치를 찾아 움직임벡터를 결정짓는 방법이다[4]. 전역탐색 알고리즘의 장점은 탐색영역 내에서 가장 적합한 움직임벡터를 구할 수 있기 때문에 영상의 화질이 우수하고 구현하기 쉽다는 장점을 가지고 있다.

이와 같이 전역탐색기법은 해당되는 모든 위치의 SAD를 계산하여 최소의 SAD를 찾아 움직임벡터를 결정짓기 때문에 탐색영역 내에서 최소 SAD를 찾을 수 있고 구현이 쉽다. 하지만 모든 블록을 계산해야 하기 때문에 다른 탐색기법들에 비해 계산범위가 많아 실시간 처리를 요하는 시스템에서의 적용은 어렵다는 단점을 가지고 있다.

ME는 전역탐색기법[5]을 제외하고도 3-Step 탐색기법, 육각탐색기법, 다이아몬드탐색기법, 빠른 전역탐색기법 등 다양한 탐색기법이 존재한다. 이러한 탐색기법들은 계산범위를 줄이기 위해 모든 블록을 탐색하기 보다는 정해진 패턴에 해당하는 블록들만을 사용하여 ME를 수행한다. 하지만 이러한 알고리즘들은 계산시간과 계산범위는 현저히 줄어들어 속도는 개선되지만 정확한 모션벡터를 예측하지 못해 화질의 저하를 가져오는 단점을 가지고 있다.

2.2 Advanced SIMD (NEON)

ARMv7 아키텍처의 선택적 구성요소인 ARM Advanced SIMD Extension이며, NEON 기술이라 부른다. 고급 미디어 및 신호처리 어플리케이션과 임베디드 프로세서를 대상으로 하는 64/128비트 복합 SIMD 기술이며, 이 기술은 ARM 코어의 일부로 구현되지만 자체 실행 파이프라인이 있고 ARM 코어 레지스터 뱅크와는 별도의 레지스터 뱅크가 존재한다. 또한 NEON은 정수, 고정 소수점 및 단-정밀도 부동소수점 SIMD 연산을 지원하고, 이러한 명령어는 Thumb-2 모드에서 사용할 수 있다.

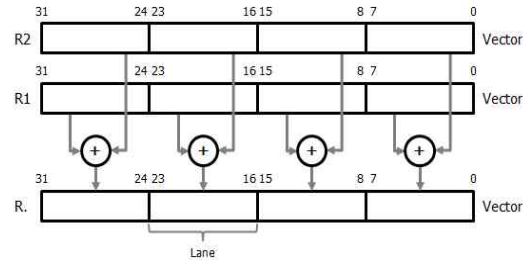

Fig. 3. Parallel NEON Operation
그림 3. 병렬 NEON 오퍼레이션

NEON은 ARM Cortex-A8부터 지원하는 Advanced SIMD 기반의 아키텍처로 오디오와 비디

오, 그래픽 처리를 위한 명령어를 추가한 미디어처리 전용엔진이다. 정수와 부동소수점을 모두 지원하며, 메모리에 직접 접근이 가능하여 효율적으로 데이터를 조작하고 성능을 향상시킬 수 있다. NEON은 그림 3과 같이 32비트 레지스터 R1, R2를 동시에 계산하여 R0에 결과 값을 저장하는 오퍼레이션을 제공한다[3]. 또한 NEON은 Core 레지스터와는 별도로 128비트 레지스터 (Q0~Q15) 16개와 32비트 레지스터 (D0~D31) 32개를 지원하여 최대 128비트의 데이터를 동시에 연산할 수 있는 성능을 지닌다[6].

2.3 Advanced SIMD (NEON) 기술사용

NEON 기술을 실제로 사용하기 위해서는 일반적인 프로그래밍 방법과는 차별화된 방법이 필요하다. NEON을 위한 어셈블리 명령어들을 처리하기 위해서 ARM의 정수 파이프라인과는 별도로 NEON을 위한 파이프라인이 존재한다. NEON을 위한 파이프라인은 NEON 어셈블리 명령어들을 수행하기 위해서 존재하지만 NEON 기술을 사용한다는 것이 NEON 어셈블리 명령어를 사용한다는 것은 아니다. NEON 기술을 사용하는 방법은 다음과 같이 3가지로 나눌 수 있다[7].

가. NEON 어셈블리 명령어

첫 번째 방법은 NEON 어셈블리 명령어를 사용하여 직접 어셈블리 코딩을 하는 방법이다. 이 방법은 가장 효과적으로 NEON 기술을 사용한다는 장점이 있지만, 기존 C/C++에 익숙해진 사용자들에게 어셈블리 프로그램을 구현한다는 것은 매우 어렵다는 단점이 있다[7].

나. NEON C언어 확장

ARM에서 제공하는 NEON C 언어 확장을 사용하는 방법이다. NEON C 언어 확장이란 기존의 C 언어 변수, 함수, 특징들과는 다르게 NEON 어셈블리 명령어들을 사용하도록 만들어진 C언어의 부가적 요소라고 할 수 있다. ARM에서 직접 제공하는 방법이기 때문에 신뢰할 수 있지만, 직접 어셈블리 명령어를 사용하는 것보다 성능이 낮다고 볼 수 없다[7].

다. NEON 자동 벡터화 컴파일러

자동 벡터화란 간단히 말하여 프로그램 코드에서 NEON 기술을 사용하여 벡터화 할 수 있는 구간을 컴파일러가 자동으로 인식하여 벡터화 시키는 방법이다. 이렇게 되면 추가적인 프로그램 변경이나 기술 획득 없이 바로 적용할 수 있다는 장점이 있지만 NEON 기술이 얼마나 적용되는지는 불분명하다[7].

2.4 Advanced SIMD (NEON) Intrinsic 함수

NEON C 언어 확장은 NEON Intrinsic 함수[3][7]로 불려진다. Advanced SIMD 명령어를 이용한 프로그램 구현은 어셈블리 언어 구조이므로 사용하기 불편한 점이 많았다. 루프 문에 대한 구현과 레지스터 개수 제한, 디버깅의 어려움 등이 따르게 되어 어셈블리 언어로 구현한 SIMD는 개발자가 모든 코드를 구현하게 되고 작성한 코드와 똑같이 CPU에서 동작하게 된다. 그렇기 때문에 개발자는 효율적인 레지스터 관리를 고려하고 불편한 디버깅과 어려운 가독성을 감안하여 프로그램을 구현해야한다. ARM에서는 이런 점들을 보완하기 위해 어셈블리 코드로 구현된 Intrinsic 함수를 제공한다. Intrinsic 함수를 이용하여 코드를 구현하게 되면 C/C++ 환경에 익숙한 루프와 조건문, 코드의 가독성, 보다 나은 디버깅 환경을 얻을 수 있고, 레지스터 개수에 대한 고민 또한 약간은 덜 수 있다[8].

Table. 1. NEON Intrinsic Function
표 1. NEON Intrinsic 함수

	Intrinsic 함수
덧셈	int8x8_t vadd_s8(int8x8_t a, int8x8_t b);
뺄셈	int8x8_t vsub_s8(int8x8_t a, int8x8_t b);
곱셈	int8x8_t vmul_s8(int8x8_t a, int8x8_t b);
...	...

NEON 기술을 실제로 사용하기 위해서는 일반적인 SIMD 명령 기반의 NEON은 어셈블리 언어구조이기 때문에 C/C++에 비해 많은 구현의 어려움이 따르게 된다. ARM사에서는 이런 단점을 보완하기위해 표 1과 같이 어셈블리코드로 이루어진 Intrinsic 함수를 제공한다. Intrinsic 함수는 C/C++ 환경에 익숙한 함수형태로 제공되며 파라미터 값만 넘겨주면 해당되는 SIMD 명령을 동일하게 수행한다. 또한 함수 호출의 오버헤드를 줄이기 위해 Intrinsic 함수는 인라인 함수로 선언되어 있다[8].

III. 본 논문의 제안방법

3.1 데이터 맵핑

본 논문에서는 다양한 ME 탐색기법 중 전역탐색 기법을 NEON에 적용하여 최소의 SAD를 계산하고 정확한 모션벡터를 구하는 방법을 제안한다. 먼저 NEON 오퍼레이션은 스칼라가 아닌 벡터방식으로 연산이 이루어진다.

그림 4의 vld1q_s16 Intrinsic 함수를 실행하면 스칼라(array_src_top, array_src_bot, array_dst_top,

```
src_top = vld1q_s16(array_src_top);
src_bot = vld1q_s16(array_src_bot);

dst_top = vld1q_s16(array_dst_top);
dst_bot = vld1q_s16(array_dst_bot);
```

Fig. 4 Mapping Intrinsic Function
그림 4. 맵핑 Intrinsic 함수

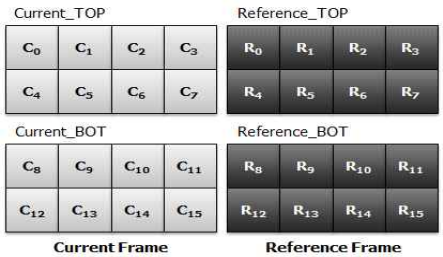

Fig. 5 4x4 Pixel Mapping Data
그림 5. 맵핑된 4x4 픽셀 데이터

array_dst_bot)였던 픽셀 데이터 8개가 그림 5와 같이 각 NEON 레지스터에 할당된 뒤 벡터 데이터(src_top, src_bot, dst_top, dst_bot)로 변환된다[9]. 이와 동일한 방법으로 8x8, 16x16 픽셀 데이터를 Intrinsic 함수를 사용하여 벡터 데이터로 맵핑시킨다.

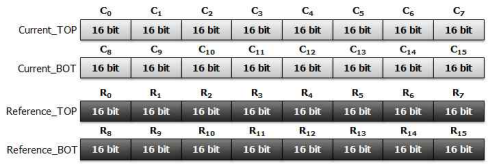

Fig. 6. NEON Register assigned to 4x4 Pixel Data
그림 6. NEON 레지스터에 할당된 4x4 픽셀 데이터

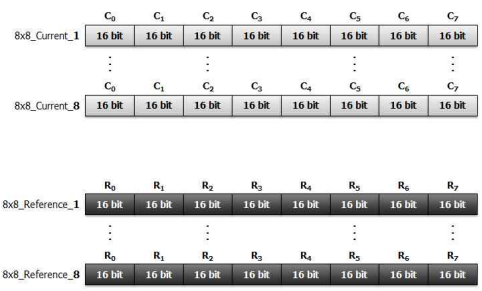

Fig. 7. NEON Register assigned to 8x8 Pixel Data
그림 7. NEON 레지스터에 할당된 8x8 픽셀 데이터

스칼라 데이터에서 벡터 데이터로 변환된 픽셀 데이터들은 그림 6, 7과 같이 각각의 NEON 레지스터에 16bit씩 할당되어 저장된 후 연산된다. 여기서 16bit씩 8개의 데이터를 저장하는 이유는 다음과 같다. NEON 기술은 동시에 128bit 연산을 지원하는 동시에 여러 형태의 레지스터 구성을 지원한다. 지원하는 레지스터의 구성은 8bit 레지스터 16개, 16bit 레지스터 8개, 32bit 레지스터 4개의 구성을 지원한다.

여기서 NEON 16bit 레지스터 8개의 구성을 선택한 이유는 영상 데이터의 표현범위 때문이다. 영상에서 1픽셀의 데이터 범위는 0~255으로 8비트면 모두 표현할 수 있지만 SAD 방정식의 특징상 탐색영역 내에 존재하는 모든 픽셀들에 대해 이전 프레임과 현재 프레임의 차이를 모두 더해야만 한다. 16비트는 표현범위가 0~65535이기 때문에 SAD 방정식에서 최악의 경우가 발생한다고 하여도 정확한 모션벡터를 구하는데 있어서 아무런 문제가 발생하지 않는다.

3.2 Intrinsic 함수를 이용한 SAD 연산

벡터 데이터로 변환된 픽셀 데이터를 전역탐색기법을 사용하여 SAD 방정식을 계산하기 위해서는 Intrinsic 뺄셈 함수(vsubq)와 덧셈 함수(vaddq)[3]를 사용하여 128bit씩 연산시킨다.

탐색영역 내에서 현재 프레임과 이전 프레임의 최소 SAD를 계산하기 위해 그림 8, 9와 같이 128비트 뺄셈 연산을 동시에 수행시킨다. 기존 전역탐색기법을 이용한 4x4 블록매칭에서는 16번, 8x8 블록매칭에서는 64번, 16x16 블록매칭에서는 256번의 연산이 순차대로 이루어지지만, NEON을 적용한 전역탐색기법에서는 NEON 오퍼레이션을 각각 2번, 8번, 32번만 사용함과 동시에 128비트 뺄셈 연산이 이루어지게 된다. 이와 같이 NEON은 기존의 순차적인 블록매칭기법 처리를 사용하는 아키텍처보다 더 빠른 시간 안

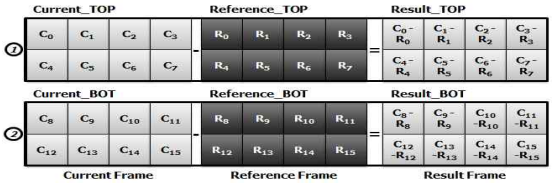

Fig. 8. 4x4 SAD Operations using NEON Subtraction Function
그림 8. NEON 뺄셈 함수를 이용한 4x4 SAD 연산

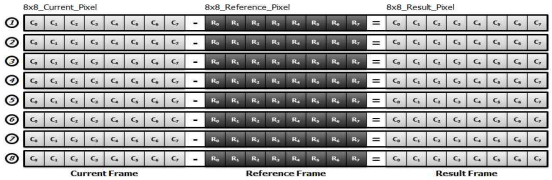

Fig. 9. 8x8 SAD Operations using NEON Subtraction Function
그림 9. NEON 뺄셈 함수를 이용한 8x8 SAD 연산

에 연산되어 처리된다.

Ⅳ. 실험 및 고찰

4.1 실험 환경

본 논문의 실험을 위해 삼성 Exynos4120 플랫폼에서 실험이 이루어졌다. Exynos4120이 탑재된 플랫폼은 Mango310 보드로써 주요 스펙은 표2와 같다.

Table. 2. Exynos4120 Hardware Specification
표 2. Exynos4120 하드웨어 스펙

Exynos4120	
CPU	ARM Cortex-A9
Memory	DDR3 SDRAM
Camera	CAM Port 2 Support
Ethernet	SMSC LAN9220

Mango310 임베디드 시스템은 ARM Cortex-A9[2]을 채택한 듀얼코어 시스템으로써, Cortex-A9은 코어별로 각각 NEON을 별도로 지원하고 있어 Cortex-A8보다 더 빠른 속도 향상을 가져올 수 있다.

본 논문에서 실험하는 방법은 NEON을 적용하지 않은 전역탐색기법과 NEON을 적용한 전역탐색기법의 비교로 진행하였다. 또한 NEON에서 제공하는 자동 벡터화 방법을 적용하여 프로세서 사이의 이식성을 유지시킬 수 있게 실험환경을 제공하였다. 실험에서 사용한 영상 데이터는 yuv sequence에서 제공하는 288p인 foreman, akiyo, mobile, news, suzie와 480p, 720p인 sildeshow, vidyo, chinaspeed 등 8개의 다양한 영상 데이터에 대해서 표본을 추출하여 약 50회에 걸쳐 반복 실험하였다.

4.2 실험 결과

본 논문에서 제안하는 NEON을 적용한 H.264/AVC ME 전역탐색기법과 NEON을 적용하지 않은 H.264/AVC ME 전역탐색기법을 수행한 결과는 표 3과 같다. NEON은 ARM 코어에서 사용하는 파이프라인과는 별도로 NEON 전용 미디어 처리 계산을 위한 파이프라인이 존재하기 때문에 순차적인 계산과 병렬적인 계산이 별도로 실행된다[2].

실험한 영상의 크기가 288p(352x288)에서 720p(1280x720)로 커질수록 제안하는 방법의 개선율이 점차 줄어들지만 NEON을 적용하지 않은 원래의 알고리즘과 비교하면 최대 65%에서 최소 0.76%의 성능 향상효과를 가져왔음을 확인하였다.

Table. 3. ME Execution Time Comparison (Unit : sec)
표 3. ME 수행시간 비교 (단위 : sec)

구분		Original Avg.	Proposed Avg.	개선율
720p	16x16	0.397441	0.367816	7.45%
	8x8	0.359871	0.334719	6.98%
	4x4	0.351947	0.349248	0.76%
480p	16x16	0.194781	0.134987	30.6%
	8x8	0.154197	0.124971	18.9%
	4x4	0.165478	0.134871	18.4%
288p	16x16	0.104815	0.037894	63.8%
	8x8	0.191576	0.041697	65.7%
	4x4	0.092182	0.033219	63.9%

Ⅴ. 결론

본 논문에서는 H.264/AVC 부호화기에서 가장 많은 연산량과 시간을 차지하는 ME를 Advanced SIMD기반의 NEON을 부분적으로 적용한 결과를 실험을 통해 알아보았다. NEON을 적용함으로써 수행시간이 영상 크기와 매크로블록 크기(4x4, 8x8, 16x16)에 따라 각각 다르게 개선됨을 확인하였다. 이러한 결과는 영상 사이즈와 매크로블록 단위가 작을수록 NEON에서 지원하는 레지스터 개수에 제한되지 않지만, 영상 사이즈와 매크로블록 단위가 커질수록 NEON에서 지원하는 레지스터 개수와 벡터 데이터로 맵핑된 레지스터 개수만큼 지원하지 않기 때문에 효율적인 계산이 이뤄지지 않아 성능이 줄어드는 결과를 나타낸다. 만약 레지스터의 개수 제한을 줄이고 효율적으로 데이터를 맵핑한다면 더 좋은 결과를 나타낼 것이다.

NEON은 PC기반의 인텔 CPU에서 지원하는 SSE2, SSE3, SSE4(Streaming SIMD Extension 4)와 동일하게 128bit 연산을 동시에 지원한다. Cortex-A8 시리즈부터 적용되기 시작한 NEON 기술은 현재 128bit 동시 연산을 지원하지만 향 후 기술발전 추세로 보아 256bit 동시 연산을 지원할 방향으로 보인다. 인텔의 하이엔드 CPU 계열인 샌디브릿지도 SIMD의 가장 최신 기술로 AVX(Advanced Vector eXtensions)[8]를 적용시켰다. AVX는 한 번에 256bit 데이터를 처리할 수 있어 128bit 동시 연산을 지원하는 SSE2, SSE3, SSE4 보다 2배 빠른 성능 향상을 기대할 수 있다. 이러한 기술적 발전으로 보아 모바일 AP(Application Processor) 시장의 선두주자로 발전해 나가고 있는 ARM사에서 이러한 추세를 반영해나갈 듯 보인다. 만약 ARM Cortex-A 시리즈에서 256bit

NEON 기술이 적용된다면 멀티미디어 응용 처리 분야에서 더욱 빠른 성능 향상 효과를 기대할 수 있을 것이다.

추후에는 DCT(Discrete Cosine Transform)와 IF(Interpolation Filter), IP(Intra Prediction) 등 H.264/AVC에서 사용되는 다양한 핵심 알고리즘들에 대해 NEON을 적용한 최적화 연구를 수행할 것이다.

References

[1] Jae-Chang Jung, *H.264/AVC Video Compress Standard*, Hong-Rling Pub, Seoul, pp.2-5, 2005.

[2] ARM, *Cortex-A9 MPCore Technical Reference Manual*, http://infocenter.arm.com/, 2010.

[3] ARM, *Introducing NEON - Development Article*, http://infocenter.arm.com/, 2009.

[4] Iain E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next generation Multimedia*, John Wiley & Sons, Hoboken, pp.32-33, 2003.

[5] Yu-Wen Hung, Bing-Yu Huang ,Tu-Chih Wang, "Analysis and Reduction of Reference Frames for Motion Estimation in MPEG-4 AVC/JVT/H.264", *Multimedia and Expo*, Vol.2, pp.II-809-12, July. 2003.

[6] Chirag Pujara et al, "H.264 Video Decoder Optimization on ARM Cortex-A8 with NEON", *INDICON 2009 Anual IEEE*, pp.1-3, Dec. 2009.

[7] ARM, *RealView Compiler Tools v3.1 - NEON Compiler*, http://infocenter.arm.com/, 2007.

[8] Young-Hoon Jung, *SIMD Parallel Programming*, Freelec, Bucheon, pp.215-544, 2012.

[9] Wing-Yee Lo, Daniel Pak-Kong Lun, Wan-Chi Siu, "Improved SIMD Architecture for High Performance Video Processors", *Circuits and System for Video Technology, IEEE Transaction on*, Vol.21, pp.1769-1783, Dec. 2011.

BIOGRAPHY

Kim Wan-Su (Student Member)

2011 : BS degree in Computer Engineering, Hanbat University.
2011 ~ Present : MS Course in Computer Engineering, Hanbat University.

Lee Jae-Heung (Member)

1983 : BS degree in Electrical Engineering, Hanyang University.
1985 : MS degree in Electrical Engineering, Hanyang University.
1994 : PhD degree in Electrical Engineering, Hanyang University.
1989 ~ Present : Professor in Dept. of Computer Engineering, Hanbat University